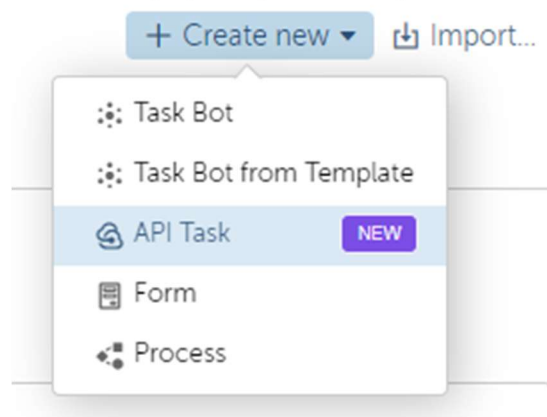


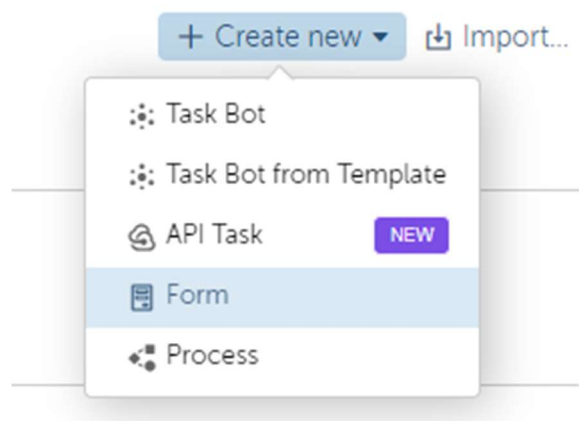
Introduction

In this exercise, we are going to put together an end-to-end automation for processing property lease documents. We are going to highlight several new technologies in the latest version of Automation Anywhere. These include:

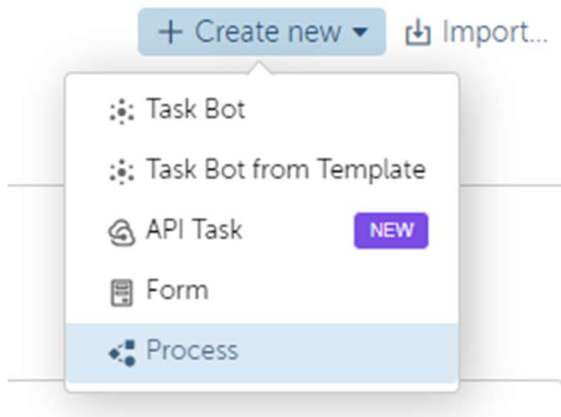
- API tasks – these are specialized automations that run in the cloud, on compartments spun up and down dynamically. When spun up, API tasks execute very quickly with response times that can be in milliseconds.



- Co-Pilot forms – these forms allow interaction with the end user. They can be used in conjunction with API tasks for fast response times and interacting with user interface elements within the form.



- Processes – these allow for easy communication between the components of the automation, from bots to forms to API tasks and more. The developer names elements for ease of use and interaction.



- Document Automation – this powerful, generative AI-powered document processing solution allows for extracting data from any document, even unstructured documents, which was thought to be impossible.

All these pieces together allow for an easy-to-use experience for users. It can also be used to mask the complexities of applications by creating our own user interface using forms and reducing manual interaction with other applications.

Getting Started

In the lab, you will see a green “Start Lab” button in the upper-left corner of the screen. Click this button to start your event.




Behind the scenes, your virtual machine is being created. When it is complete, you will have full access to a Windows virtual machine. You will complete your lab on this virtual machine, within your browser, not on your regular computer. Because this is running on a virtual machine, you can access this with any Internet-connected computer. Once your virtual machine is ready, you will see 5 fields of data on the left side of the screen, you will need these later on! This is the starting vm screen. They contain:

- The web address of your RDP session/Automation Anywhere control room (this is accessed through a browser, not a standard remote desktop client)
- The Automation Anywhere username
- The Automation Anywhere password for the Control Room
- And the Google Cloud API key, used for the exercise


End Lab **01:55:49**

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)


Automation Anywhere Client




Automation Anywhere Username



Automation Anywhere Password



Google Cloud API Key



Use the Copy icon next to the AA Remote RDP Session field, then open a new tab in your browser, paste the URL into the address line, and press ENTER. Note: You will get a warning that the connection is not private. This is normal. On this screen, click Advanced, then click on the “Proceed to xx.xx.xx.xx (unsafe)” link. This is because we do not have a certificate for this dynamic VM and is no cause for alarm.



Your connection is not private

Attackers might be trying to steal your information from **34.27.151.151** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID



To get Chrome's highest level of security, [turn on enhanced protection](#)

Hide advanced



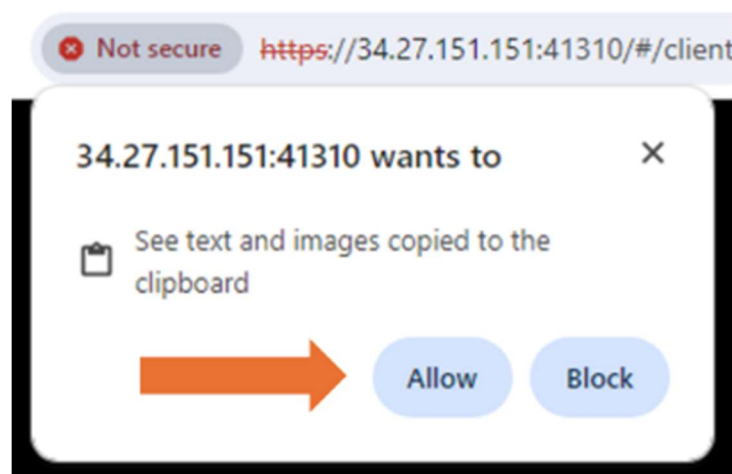
Back to safety

This server could not prove that it is **34.27.151.151**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to 34.27.151.151 \(unsafe\)](#)



A pop-up will appear asking for clipboard access. Click the Allow button. You now have full access to your Windows virtual machine.



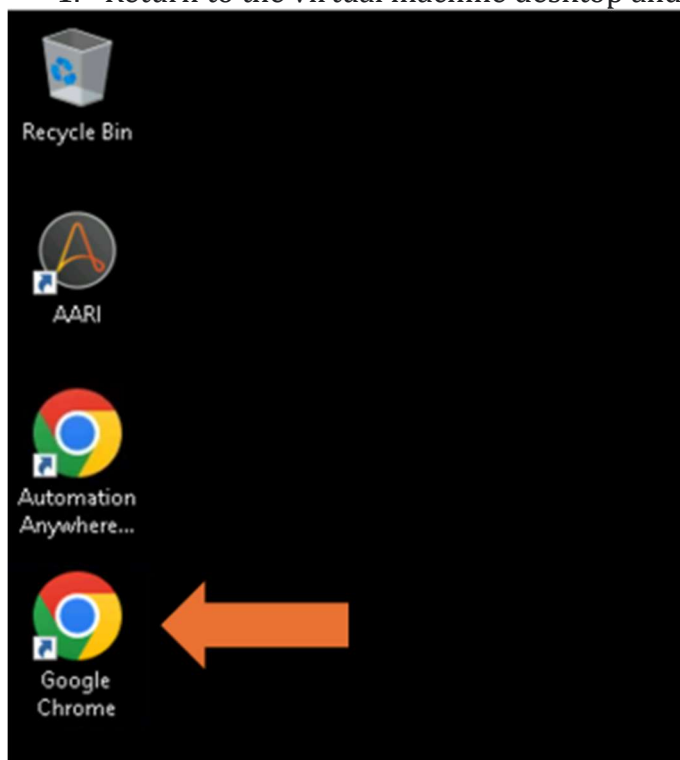
Launching the Control Room

In the virtual machine, there is a Chrome link on the desktop titled “Automation Anywhere Control Room”. Double-click this to launch the Control Room.

- For the username, switch back to the first tab where the lab was launched, and copy the Automation Anywhere Username, then switch back and paste that into the Username field.
- For the password, switch back to the first tab where the lab was launched, and copy the Automation Anywhere Password, then switch back and paste that into the Password field.
- You may then click Log in.

To get the partially completed lab files, follow the following steps:

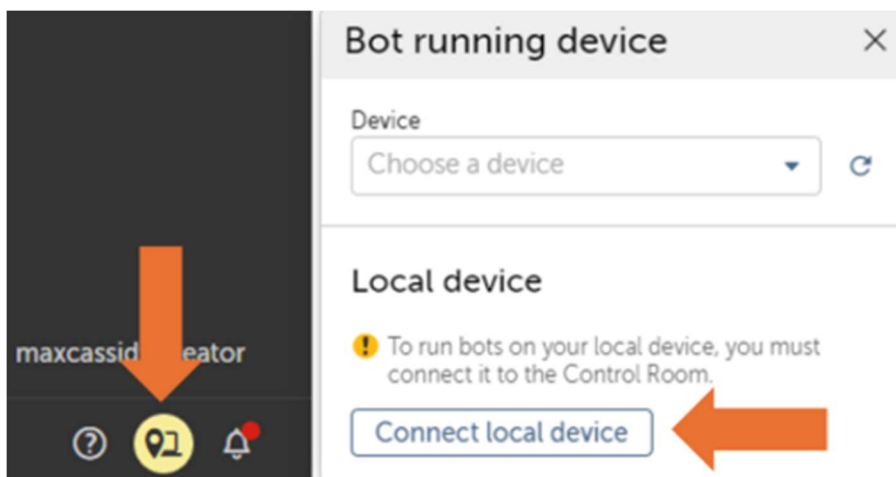
1. Return to the virtual machine desktop and open Google Chrome



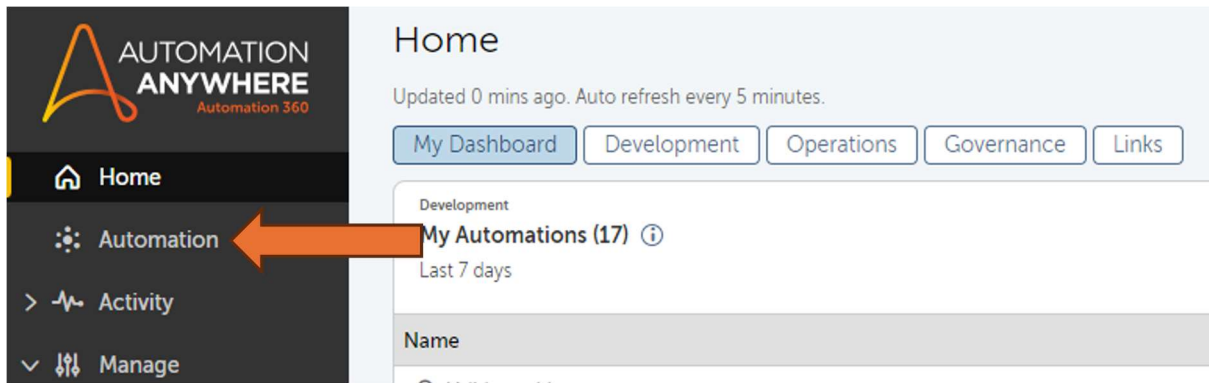
2. To download the files, navigate to: Insert link here!!!!
3. Once downloaded, close Google Chrome, NOT your VM, and open Automation Anywhere Chrome icon



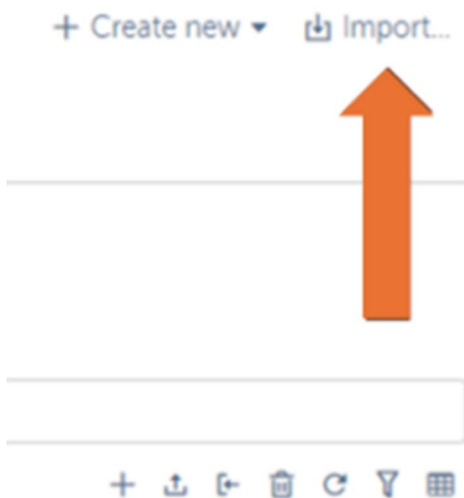
4. Enter your username and password that is captured from the starting VM screen we saw earlier.
5. In the bottom left of your control room once logged in, you will see an image of a computer, click this and then press connect local device.



6. A pop up will appear, from here press the connect button, and wait for it to finish. When it is connected and the done button appears, click Done.
7. Click on Automation on the left-side menu.



8. Then, from the upper-right corner, click Import.



9. Click the Browse button on the right side. A file open dialog appears. In the quick access pane on the left hand side, select "Downloads", Select the Client Onboarding Lab Zip and click Open, make sure you HAVE NOT unzipped the file before selecting it.
10. Before pressing Import Bots, make sure to **SELECT SKIP BOT FILE WHEN UPLOADING**

Import bots

To import bots, please select the file that you exported from the Control Room.

Where is the file you want to import

GoogleNextLab.zip

Password (optional)

This import may require a password. Check with the person who exported this file.

Import bot to

☐ Public tab

☒ Private tab

During import, if a file already exists

☒ Skip the bot or the file (don't import it)

☐ Overwrite the bot or the file with the imported one

11. Then you can click Import bots in the upper-right corner of the screen. This will return you to the Automation screen. You should see a new folder titled "Client Onboarding Lab" in your private repository. You may have to press the refresh (↻) button above the file listing to see the folder.

The screenshot shows the Automation ANYWHERE interface. On the left is a dark sidebar with the logo and navigation menu. The main area is titled 'Automation' and has tabs for 'Public' and 'Private'. The 'Private' tab is selected. Below the tabs, a message states: 'Private bots and files cannot be viewed by other people. If a bot or file has been checked out from the Public tab, it can be viewed and run by other people, but cannot be edited.' There are two sections: 'Folders' and 'Files and folders (6)'. The 'Folders' section lists several folders, with 'Google N...' (Google Next 2024 Lab) highlighted. The 'Files and folders (6)' section shows a table of items.

	Type	Name	Status
<input type="checkbox"/>	Task Bot	Enter data to webpage	New
<input type="checkbox"/>	Task Bot	Extract document information	New
<input type="checkbox"/>	Form	Input form	New
<input type="checkbox"/>	Process	Lease document process	New
<input type="checkbox"/>	Task Bot	Process file	New
<input type="checkbox"/>	API Task	Validate address	New

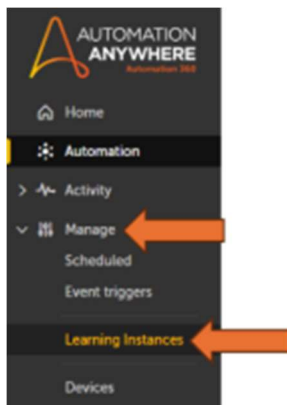
If you click on the Client Onboarding Lab folder, you will see component pieces of this automation: Three task bots, a form, a process, and an API task, just like the image above. We will be putting some finishing touches on some of these files and then testing them with some sample lease documents that we will download later.

Creating the Learning Instance

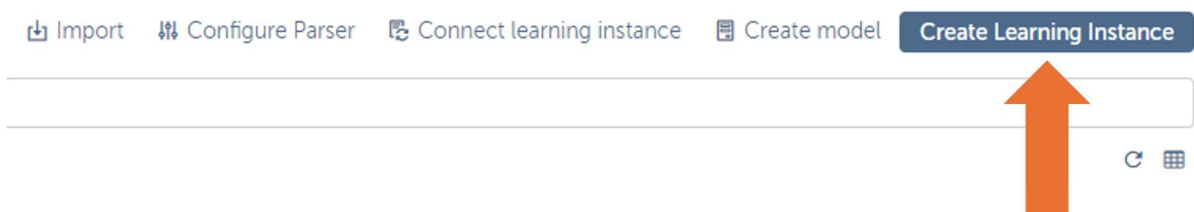
Before we start modifying the component pieces of the lab, we need to create a “Learning Instance”. This defines the type of document we wish to use, the fields and tables that we want to extract, and rules regarding the data that is extracted from the document.

In our case, we are processing rental lease agreements. These documents do not present information in a structured fashion, making them difficult or even impossible to process using standard document processing applications.

By utilizing the power of generative AI, we can extract meaningful information from the documents. Using some proprietary technologies, we can do it faster than just feeding the document to a generative AI. From the left-side menu, choose Manage > Learning Instances.



You can ignore existing learning instances because we want to create our own from scratch. Click the Create Learning Instance button in the upper-right corner of the screen.



For the name, enter “Google Cloud” followed by your full name. The names of the learning instances must be unique. You may put in an optional description.

Create Learning Instance

[Cancel](#)[Next](#)

Name

Google Cloud - Aaron Gleason

Description (optional)

Document Type

Unstructured document

Language

English

Locale

English (United States)

Provider

Automation Anywhere (User-defined)

OCR Provider

Google Vision

☒ Generative AI-driven data extraction

Your document data will be processed by an AI language model. [learn more...](#)

For the Document Type, select Unstructured Document. Then we will leave the rest of the fields as they are: Language, Locale, Provider, and OCR Provider. This will utilize the Optical Character Recognition (OCR) capabilities of Google Vision to read the document. At this point, just press Next.

When the pop-up window appears, click Add a field to get started. For the Field name, enter Landlord Name. Add the same thing for the Field label. Sometimes the field label will be different based on how the document is formatted.

If you tab away from the Field label text box or click outside the text box, you will see the query for generative AI prompt, this will be: "What is the Landlord Name?"

Google Cloud - Aaron Gleason

[Cancel](#)[Create](#)

Form fields

Select required and optional fields for extraction.

Field name

Search

<input checked="" type="checkbox"/>	Field name	Field label	Data type
<input checked="" type="checkbox"/>	Landlord Name	Landlord Name	Text

Show unused fields

Add a field

Field Properties Field Rules Document Rules

Field name

Landlord Name

Field label

Landlord Name

Confidence (0-100)

0

+

-

Data type

Text

Required

☒ Required

☐ Optional

☒ Search query for Generative AI model

Language model will extract data using the provided search query.

What is the Landlord Name?

Should your document require it, you may add additional text to the prompt to be more specific about your needs, such as specifying the type of document it is processing or the specific format in which you need the returned data. This can add context to the generative AI's extraction process.

Click the Add a field button to add the following fields in the same fashion:

- Tenant Name

- Property Address

For our next field name and field label, we'll want to adjust the prompt a bit to add some context and specify the expected responses.

Add your field label as Type of Lease:

- Type of Lease

Once you've added the Type of Lease field name and field label, we'll want to adjust the prompt. In your prompt input, after "What is the Type of Lease?" – add the text "Respond with either RESIDENTIAL or COMMERCIAL".

Your prompt should be: "What is the Type of Lease? Respond with either RESIDENTIAL or COMMERCIAL."

Finally, we will have to configure this slightly, but to start click the Add a field button to add the following, final, field:

- Monthly Rent

Monthly Rent will require a slight change in the field settings. First, change the Data Type to Number.

After the "What is the Monthly Rent?" prompt, add, "Return the total rent as a regular, unformatted number."

Your prompt should be: "What is the Monthly Rent? Return the total rent as a regular, unformatted number."

After completing these fields, click the Create button at the top right.

Form fields

Select required and optional fields for extraction.

Field name

Search

<input checked="" type="checkbox"/>	Field name	Field label	Data type
<input checked="" type="checkbox"/>	Landlord Name	Landlord Name	Text
<input checked="" type="checkbox"/>	Tenant Name	Tenant Name	Text
<input checked="" type="checkbox"/>	Property Address	Property Address	Text
<input checked="" type="checkbox"/>	Type of Lease	Type of Lease	Text
<input checked="" type="checkbox"/>	Monthly Rent	Monthly Rent	Number

Show unused fields

Add a field

Field Properties

Field Rules

Document Rules

Field name

Monthly Rent

Field label

Monthly Rent

Confidence (0-100)

0

+

-

Data type

Number

☐ Format Number

Updates the appearance of extracted numbers based on your locale

Required

☒ Required

☐ Optional

☒ Search query for Generative AI model

Language model will extract data using the provided search query

What is the Monthly Rent?

Integrating the Learning Instance

Process File

We now shift to the automation side of the house. Click on Automation on the left-side menu. Inside, click on the Client Onboarding Lab folder (if you haven't already), then click on the Process file task bot.

Folders

- ▼ Bots
 - COMMON
 - CoPilot GenAi test
 - > Document Workspace P...
 - > Events
 - GCP Demo Single Docu...
 - Google N...**
 - Sample bots
 - > Update Videos
 - Bot Store

☐ Search within subfolders

Name
Search

Files and folders (6)

<input type="checkbox"/>	Type 1	Name 2	Status
<input type="checkbox"/>	Task Bot	Enter data to webpage	New
<input type="checkbox"/>	Task Bot	Extract document information	New
<input type="checkbox"/>	Form	Input form	New
<input type="checkbox"/>	Process	Lease document process	New
<input type="checkbox"/>	Task Bot	Process file	New
<input type="checkbox"/>	API Task	Validate address	New

This taskbot consists of 4 actions, we need to click once on the Extract data action on line 3 and the parameters will appear on the right side of the screen. (If you accidentally double-clicked, just click once again.) On the right side under Learning Instance Name, to the right of the drop down is a refresh icon, (↻), click this. Then, click the drop-down menu and choose your “Google Cloud [your name]” entry.

Start

1. Step "Process file"

2. Error handler: Try

3. Document Extraction: Extract...

4. Error handler: Catch AllErrors

Drag a trigger here...

Drag an action here...

Document Extraction: Extract data

Processes uploaded documents to extract data
Required bot agent version: 21.98 or above

Document to extract
Variable Control Room file Desktop file

Regex support unavailable. Upgrade package to enable regex.

Learning instance name
Server instance Variable

Output results should be
☒ Uploaded to server
☐ Saved to a local folder

Output folder path
Variable Desktop folder

Browse...

Path to local file system


Service account (optional)
Credential Variable Insecure string

Pick...

Service account from Google. Optional parameter for using your own Google service account to access Doc AI.

Document AI endpoint URL for document processor (optional)


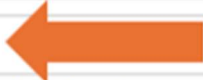

This action works by receiving an input file from a form that we created, specifically the \$fInputFile\$ under Document to extract.

Document Extraction: Extract data 

Processes uploaded documents to extract data
Required bot agent version: 21.98 or above

Document to extract

Variable Control Room file Desktop file

 **\$fInputFile\$**  

Regex support unavailable. Upgrade package to enable regex.

Generative AI then kicks in to process the document based on the learning instance you defined. By passing our prompt, we are able to capture the required data based on a question.

Click Save, then Close in the upper right corner.

Extract Document Information

After the documents are processed, we need to retrieve the data from the server. We do this by using the Extract document information task bot.

Folders

- Bots
 - COMMON
 - CoPilot GenAi test
 - Document Workspace P...
 - Events
 - GCP Demo Single Docu...
 - Google N...**
 - Sample bots
 - Update Videos
 - Bot Store

☐ Search within subfolders

Name
Search

Files and folders (6)

<input type="checkbox"/>	Type 11	Name 12	Status
<input type="checkbox"/>	Task Bot	Enter data to webpage	New
<input type="checkbox"/>	Task Bot	Extract document information	
<input type="checkbox"/>	Form	Input form	New
<input type="checkbox"/>	Process	Lease document process	New
<input type="checkbox"/>	Task Bot	Process file	New
<input type="checkbox"/>	API Task	Validate address	New

In this taskbot, the action on line 3, Document Extraction: Get document data action retrieves the extracted data from the server using the document ID. The ID was generated when we submitted the file for processing and then was made available for other actions by marking the document ID variable as “input”.

Extract document information

Variables

+
:

Search variables

Your variables

- rReturn
- sDocumentID**
- sErrorMessage
- sExtractionStatus
- sLandlordName

Triggers

Start

Step "G

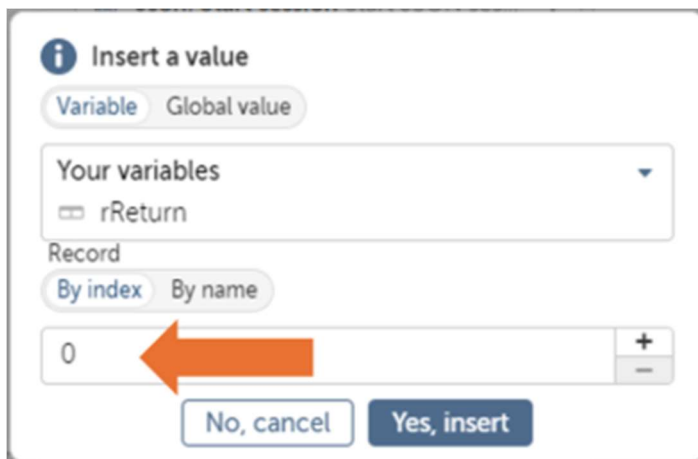
Error

When getting the document info from the Document Extraction: Get document data action, five columns of indexed data are returned and stored within the record variable called rReturn.



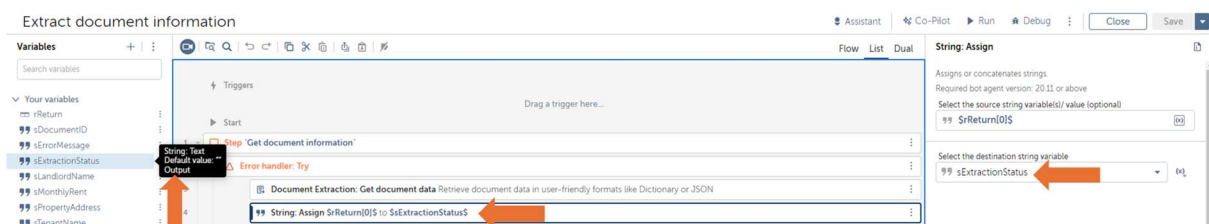
The type of data returned depends on the index we select:

- Document extraction success message (index 0)
- Document data download success message (index 1)
- A dictionary data format containing the tables, pages, and fields (index 2)
- A JSON message containing the tables, pages, and fields (index 3)
- The document ID (index 4)



We store these into a Record variable to extract what we need. For example, the String: Assign action takes the extraction success message and assigns it to a string variable.

And as seen below, we have marked our variable as an output, so we know if the document should be sent to validation.



After that, we process the JSON message to extract the field values.

We use the action JSON: Get node value to accomplish this. The field names are the same as we created in the learning instance. Since we specified spaces in the fields, we need to contain the field names in brackets and apostrophes like this: ['Property Address']

The screenshot shows a workflow editor with a sequence of actions. An orange arrow points to the 'Json: Get node value' action at step 6. The right sidebar shows the configuration for this action, including the JSON node key path '\$\$.fields.['Landlord Name'].value', session name 'Default', and the variable 'sLandlordName'.

The “JSON: Get node value” action uses JSON node notation. Here are the nodes we have for you. You must add the missing fields to this task bot.

Field name	JSON node notation	Variable name
Landlord Name	fields.['Landlord Name'].value	sLandlordName
Monthly Rent	fields.['Monthly Rent'].value	sMonthlyRent
Property Address	fields.['Property Address'].value	sPropertyAddress

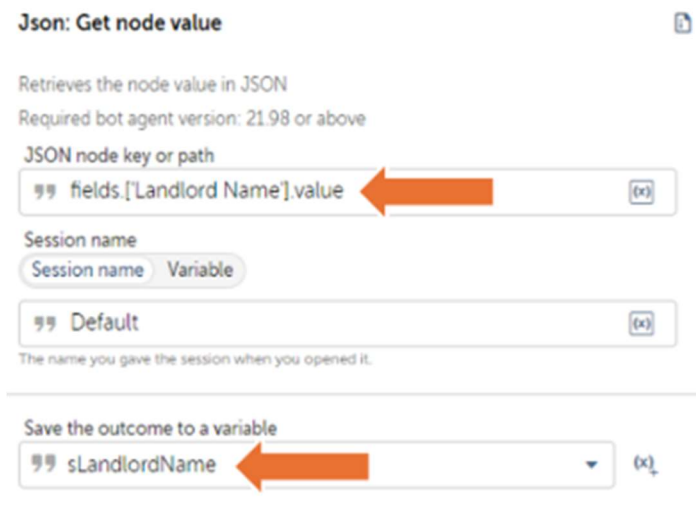
Your task is to add more "JSON: Get node values" for the following fields. The variables are already created for you.

Hint: You can copy and paste existing actions by selecting the 3 dots to the right of the action or using keyboard shortcuts.

Field name	JSON node notation	Variable name
Tenant Name	fields.['Tenant Name'].value	sTenantName

Type of Lease	fields.['Type of Lease'].value	sTypeOfLease
---------------	--------------------------------	--------------

Note: Be sure to add the Get node value actions before the JSON: End session action.



Json: Get node value

Retrieves the node value in JSON
Required bot agent version: 21.98 or above

JSON node key or path
fields.['Landlord Name'].value

Session name
Session name Variable
Default

The name you gave the session when you opened it.

Save the outcome to a variable
sLandlordName

After those additions, Save then Close this task bot.

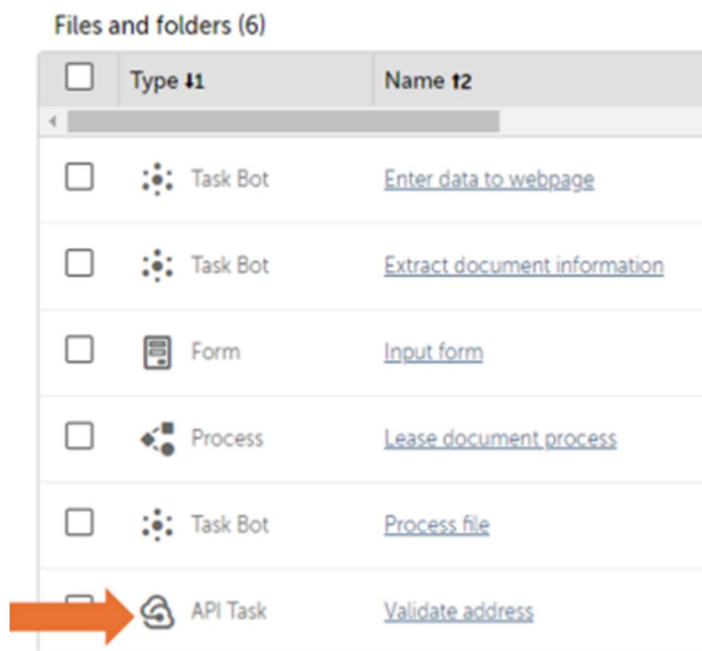
Validate Address

When we extracted the data from our lease agreements, we extracted the address. However, addresses can have discrepancies in formatting. To avoid this problem, we'll leverage an API task to utilize Google Map's address validation API to standardize our extracted address format.

What's the difference between an API task and a regular task bot? An API task runs on the cloud, when executed, this API task triggers a virtualized environment containing all necessary components to complete the task, this is known as a container and is spun up in real-time for this functionality. It does not run on the client workstation, some advantages of this are:

- It runs on server-class machines, so processing is fast. It spins up and down the container on demand, reducing cloud server resources.
- Once spun up, the API task can be called multiple times with very fast response rates.
- Since API tasks run in the cloud rather than the local workstation, they can be called from anywhere – in a process, in a form, or via API.

- They are small and can be built function specific – great for reusability.



Our API task, Validate Address, is a simple one: We start by calling the Google Map's API, we then send the address we extracted from the document and return the corrected address.

API tasks can execute most non-UI manipulating tasks including connecting to web-accessible databases, Google Office and Microsoft Office 365, and more.

Notice that we are using a `$$sKey$$` variable in the URI of the REST web service.



This variable contains the API key that was entered in the initial form. We will utilize the Process to map the value from the form to the variable in the API task.

REST Web Services: Post method



Creates a new resource in the URI. Parameters are passed in the request body, and there is no limit on the length for a request body.

Required bot agent version: 20.11 or above

URI

Enter the URI

https://addressvalidation.googleapis.com/v1:validateAddress?key=\$sKey\$

(x)

e.g https://domainname.com/resource/search?param1=value1¶m2=value2



Now, underneath the Json: Start Session action, we need to add a get node. In the left hand actions bar, find Json: Get Node Value, and drag this to line 5, ensuring it is inbetween Json: Start Session and Json: End Session.

In the parameters on the right hand side, in the JSON node key or path input box, enter: `$$result.address.formattedAddress`

Now, at the bottom of the parameters, select the dropdown under Save the outcome to a variable, and select: `sPropertyAddress`

Json: Get node value



Retrieves the node value in JSON

Required bot agent version: 21.98 or above

JSON node key or path

`$$result.address.formattedAddress`

(x)

Session name

Session name Variable

Default

(x)

The name you gave the session when you opened it.

Save the outcome to a variable

`sPropertyAddress`

(x)

Now press save and close at the top right.

Input Form




Speaking of forms, let's complete our form. Open the Input form.

Files and folders (6)

<input type="checkbox"/>	Type #1	Name #2
<input type="checkbox"/>	Task Bot	Enter data to webpage
<input type="checkbox"/>	Task Bot	Extract document information
<input type="checkbox"/>	Form	Input form
<input type="checkbox"/>	Process	Lease document process

We only have some instructions and a link to the exercise web page in the form. We need to add two elements: The "Select File" and a "Text Box" for the API key.



Input form

Form

Layout

- Column

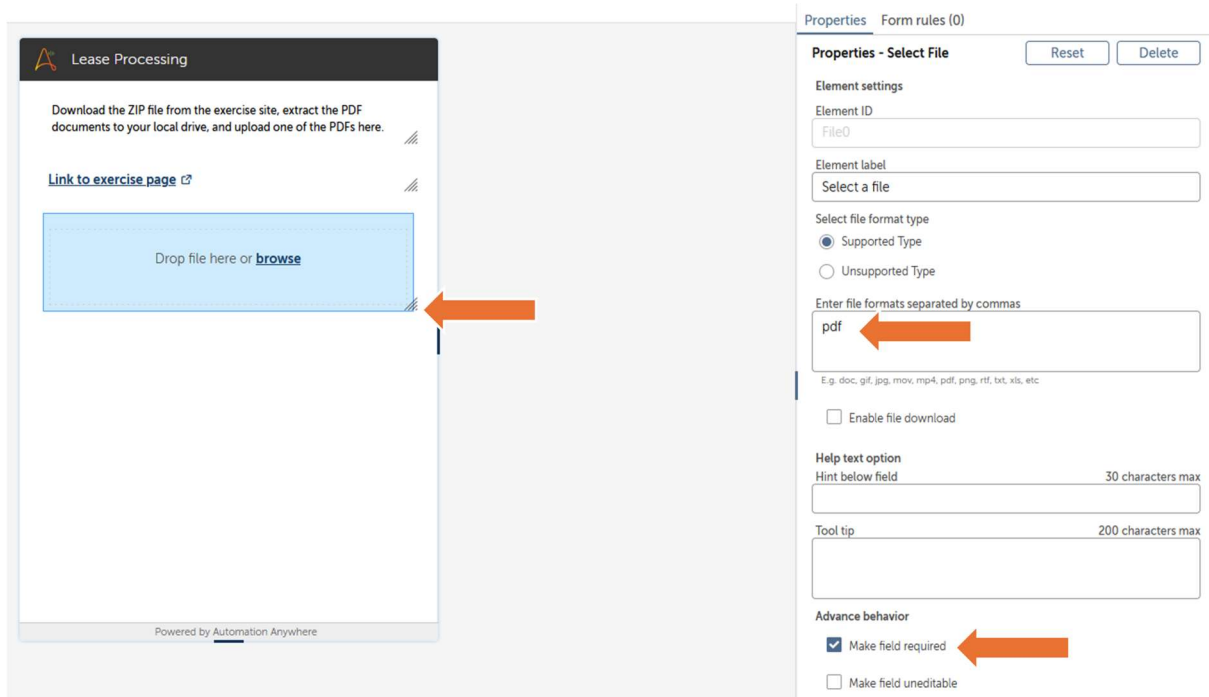
Elements

- BlankSpace
- Button
- Checkbox
- Date
- Document
- Dropdown
- Dropdown Multi-Select
- Dynamic Area
- Hyperlink
- Image
- Label
- Number
- Password
- Radio Button
- Rich Text Editor
- Select File 
- Select Folder
- Snapshot
- Table
- Text Area
- Text Box 
- Time

Let's start with the Select File. Drag Select File from the left-side menu into the form below the link. You may resize the element to the width of the form by dragging the bottom-right corner to the right.

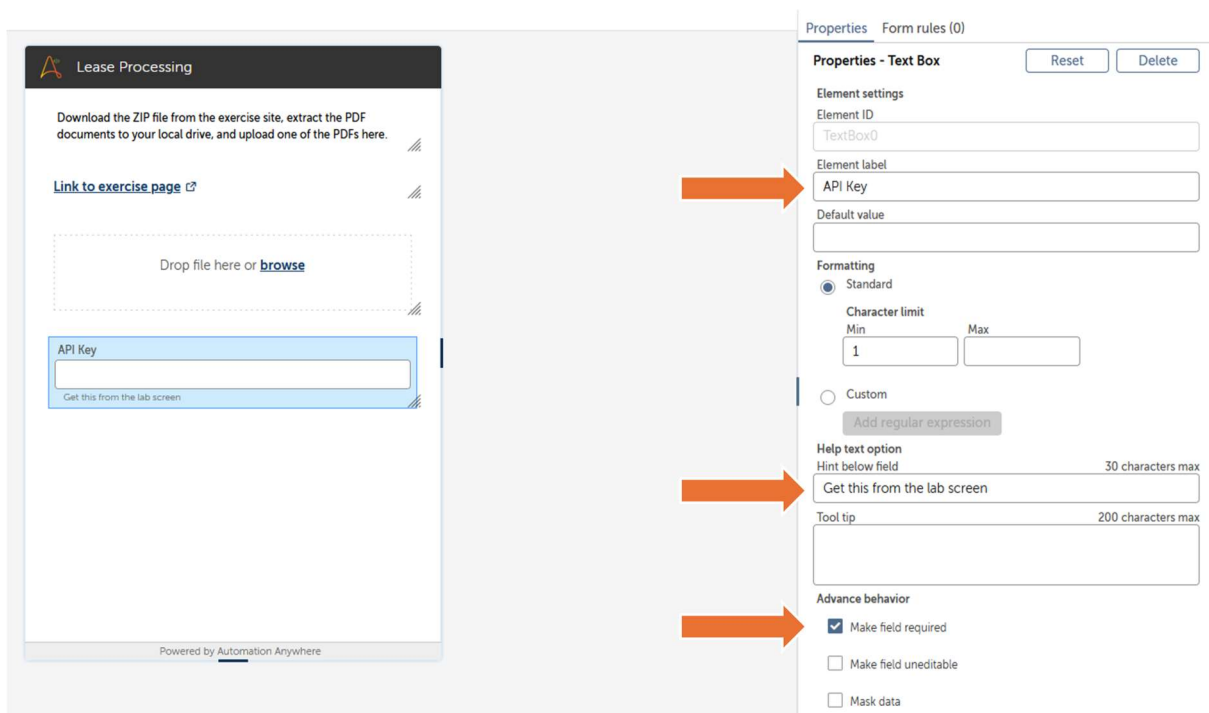
On the right side of the screen are the properties of this element. The element ID is automatically generated and cannot be changed. If you wish to change the label, you may do so.

In the text area for "Enter file formats separated by commas", enter pdf. You may add a hint or tool tip if you wish and click the box for "Make field required".



Next, drag the Text Box element from the left-side menu into the form, below the Select File element. You may also resize the text box. On the right-side properties menu.

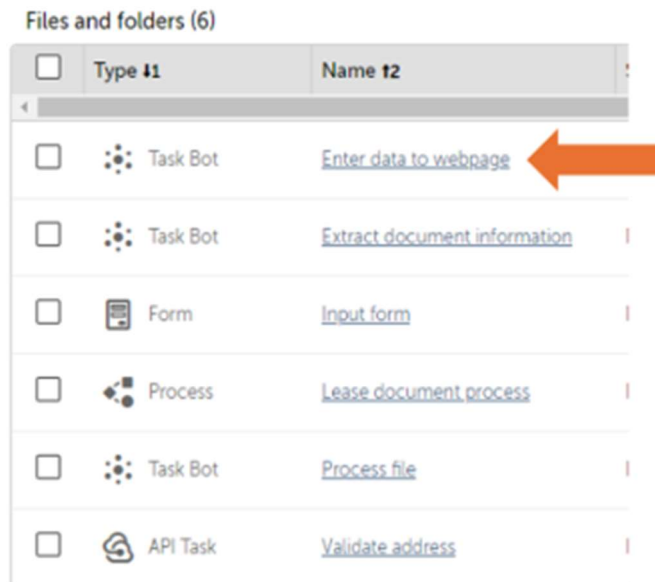
Change the Element label to “API Key”. In the “Hint below field”, enter “Get this from the lab screen”. You may also check the box for “Make field required”.



Save your changes and Close the form.

Enter Data To Webpage

Let's look at the Enter data to webpage task bot.



This launches a fresh copy of the lease entry web page and enters data into the form.

Speaking of which, open a separate Chrome window in your virtual machine and launch the lease entry page at this address:

<https://www.google.com>

Below the "Browser: Open" action, which opens the fresh copy of the web page, are a bunch of Recorder: Capture actions. These are the actions that enable the interaction with a user interface.

Look through each of these actions, we currently have:

- Landlord Name
- Tenant Name
- Property Address

We still need the following:


- Monthly Rent
- Type of Lease
- The Terms and Conditions radio button

- The Submit button

Your task is to add the additional "Recorder: Capture" actions for the remaining data fields above.

You may copy and paste the Recorder: Capture actions to create new ones. When you do, use the button labeled "Recapture object".


When you do, the lease entry page comes forward and you can select the appropriate field.

Recorder: Capture 


Record interactions with UI elements such as click, read, and write.
Required bot agent version: 21.210 or above

Window

☐ Browser ☐ Application ☐ Variable

☐ \$Browser1\$ 

Browser title

 Property Management Client Onboarding Challenge

☒ Case sensitive


Link of web page

Browser application

Google Chrome

☐ Resize window
May improve bot accuracy

Main Anchor

Recapture object 

Previous

When capturing an object with the Recorder: Capture action, when you move your mouse around the web page, you will notice it highlighting fields in red.

To select that field, just single-click on the field once it's highlighted.

Lease Processing
Add the details of each new lease below to ensure all lease are in the system of record.

Lease Details

Landlord Name Tenant Name

Property Address

Monthly Rent Type of Lease

Please fill out this field.

Submit Lease
By submitting the lease above, I acknowledge that the lease details are true to the best of my knowledge. Once the lease is submitted, it will go through a review process before final approval.
Please acknowledge that you agree to the terms above:
☐ Yes ☐ No

Submit

©2024 Automation Anywhere, Inc. Find more hands-on automation training: <https://pathfinder.automationanywhere.com/>

For the Monthly Rent field, scroll down to “Enter keystrokes here or use the on-screen keyboard”.

Remove the variable that’s there (e.g., sPropertyAddress), then click the X in parenthesis at the end of the text box. Choose sMonthlyRent from the list and click Yes, insert.

Action to take on object

Set text

☐ Run in background
A person who is logged in on the device can perform tasks while this action takes place

Keystrokes

☒ Enter keystrokes here or use the on-screen keyboard

” \$sMonthlyRent\$ (x)

Make another copy for the Type of Lease.

Click Recapture object and choose the appropriate field in the web page.

Scroll down to “Action to take on object” and change that to “Select item by text”.

For the Assign Value just below that, click the X in parenthesis and choose the sTypeOfLease variable.

Action to take on object
Select item by text

☐ Run in background
A person who is logged in on the device can perform tasks while this action takes place

Assign value
\$sTypeOfLease\$

Make another copy for the terms and conditions radio button.

Click Recapture object and click the radio button in front of the word “Yes” on the web page.

Scroll down to “Action to take on object” and choose “Select”.

Preview

Please acknowledge that you agree to the terms and conditions

☒ Yes ☐ No

Action to take on object
Select

One last copy for the Submit button.

Click Recapture object and click on the Submit button.

Scroll down to “Action to take on object” and choose “Click”.

Preview

Submit

Action to take on object
Click

Save and Close the task bot.

Lease Document Process

Now to complete the Process and test our automation.

Processes are ways of encompassing multiple elements, such as task bots, API tasks, forms, and more, making them work together and share data.


Could this entire automation have been completed using a single task bot? Yes!

Would that be a good segmentation of code allowing for reuse? No!

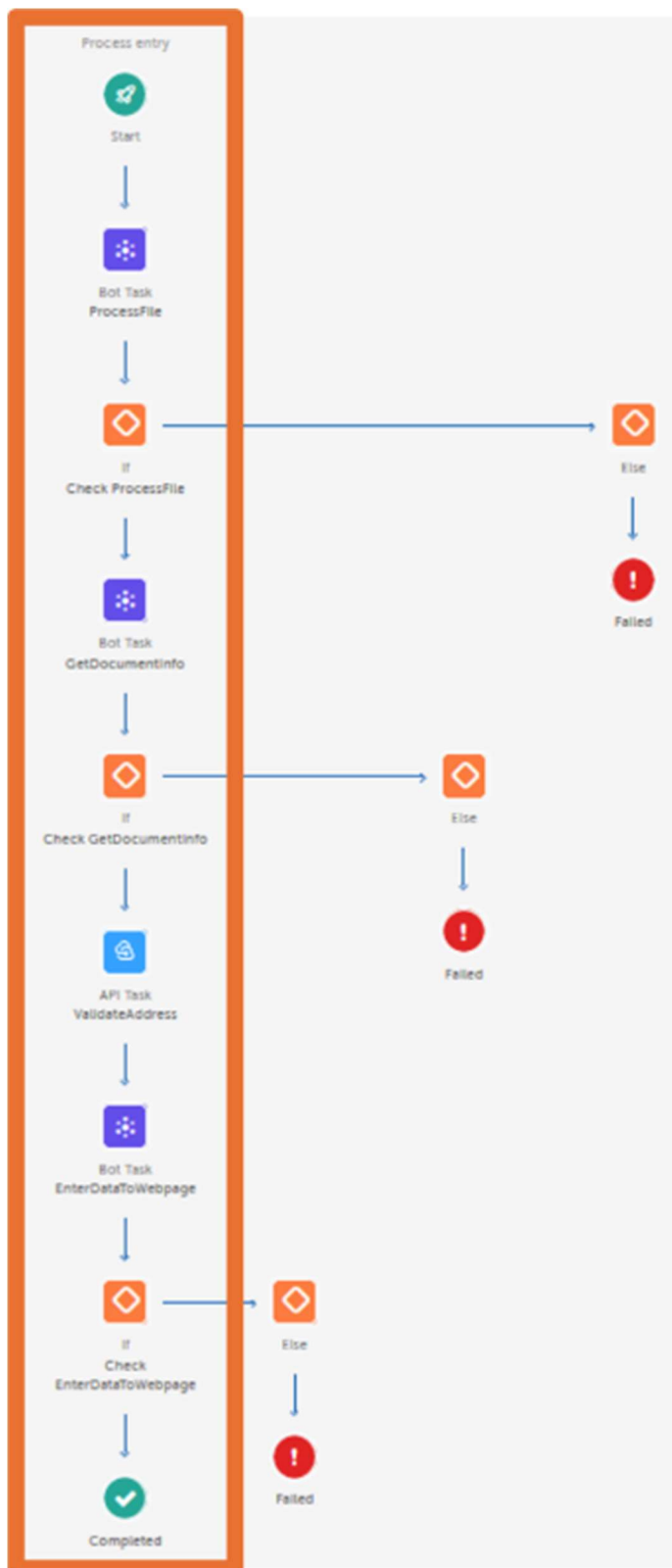
Let's open the Lease document process.

Files and folders (6)

<input type="checkbox"/>	Type #1	Name #2	
<input type="checkbox"/>	Task Bot	Enter data to webpage	
<input type="checkbox"/>	Task Bot	Extract document information	
<input type="checkbox"/>	Form	Input form	
<input type="checkbox"/>	Process	Lease document process	
<input type="checkbox"/>	Task Bot	Process file	
<input type="checkbox"/>	API Task	Validate address	



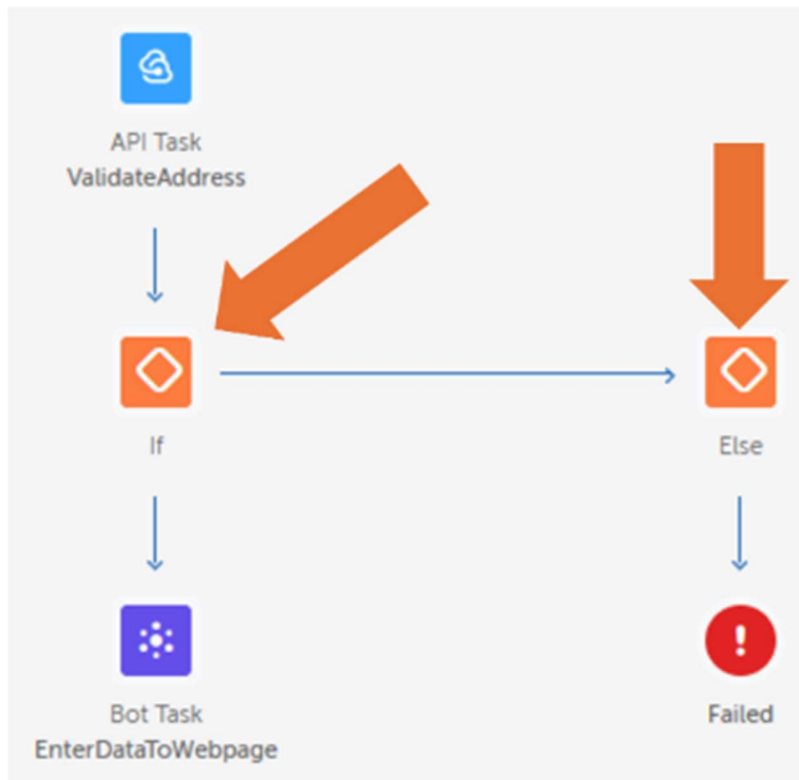
Right now, it goes from Start to End in a straight line. That means, if a document is unable to be confidently read, there is no facility to go to document validation. We need to add that functionality.



Each node in a Process can receive or share a value by using variables marked as “input” within the automations and share a value using variables marked as “output”.

One point of contention is that multiple nodes can have the same output variables. They will only be differentiated by the node name, so be certain to use node names that accurately describe the node.

To start, add an “If/Else pair” after the API Task – ValidateAddress node. This is where we will check the validation status of the document.



For the Description, enter ValidationStatus.

In the Condition, this will be a String condition.

For the Source value, click on the X in parentheses, under Task, choose GetDocumentInfo. The variable type is Output and choose GetDocumentInfo.sExtractionStatus from the list and click Add.

Then select the sExtractionStatus variable from the list and click Add.

Notice the format of the variable: GetDocumentInfo.output[sExtractionStatus] It is the node name, and “output” along with the variable in brackets.

For the Operator, select Includes from the list.

This will look anywhere within the string for the value we want. Set the Target Value to SUCCESS.

Now, the first part of our IF is settled. If the sExtractionStatus includes (or contains) the word SUCCESS, the Process will continue to the EnterDataToWebpage node. Otherwise, it will follow the Else path.

Condition: If

Runs a sequence of actions if the conditions are true

Description (optional)
ValidationStatus

Used only as internal reference during design

Update task name (optional)
ProcessFile (x)

Display message (optional)
(x)

Displayed as a note inside the request

Condition
String condition

Checks the string variable condition

Source value (optional)
\$GetDocumentInfo.output[sExtractionStatus]\$ (x)

Operator
Includes

Target value (optional)
SUCCESS (x)

☒ Match case

Add Condition

After the Else, add the Document Validation element from the left-side menu.

For both the Element name and the Task name, enter "ValidateDocument".

For the Document ID, click the X in parentheses and, under Task, choose ProcessFile.

Variable type is Output.

Under Variable, choose ProcessFile.sDocumentID.

Task: Document Validation

Displays documents that require validation

Element name

ValidateDocument

Used only as internal reference during design and must be unique within this process

Task name

ValidateDocument

(x)

☐ Hide this task after completion

Task requester and assignment

Default

Custom

The assigned group of the request will be responsible for completing this task. By default, this task will only be shown in the "Assigned" task list.

☐ Show this task in "Requested" task list for the request creator

Task contributor

Manually assigned

Document ID

\$ProcessFile.output[sDocumentID]\$

(x)

Data privacy tag (optional)

(x)

Used only as an additional hidden field that can be searched for Data Privacy purposes.
Variables will be populated at runtime

After the Document Validation node, add a Bot Task node.

Since changes could have been made during the document validation process, we need to re-retrieve the data from the document.

To start, for both the Element name and the Task name, enter "GetDocumentInfoAgain" so as not to conflict with the original document info retrieval node.

Next, to the right of Select Bot, press on browse and a pop-up will appear.

Select browse in the top right and select the automation called "Extract document information".

For the Input values, check sDocumentID then click the X in parentheses.

Under Task, choose ProcessFile, then select Output and select the variable ProcessFile.sDocumentID.

Task: Bot

Executes a bot with inputs and outputs

The screenshot shows the configuration interface for a 'Task: Bot'. It includes several input fields and controls:

- Element name:** A text box containing 'GetDocumentInfoAgain'. An orange arrow points to this field.
- Task name:** A text box containing 'GetDocumentInfoAgain' with a '(x)' icon to its right. An orange arrow points to this field.
- Hide this task from users:** An unchecked checkbox.
- Bot queue timeout:** Two input boxes for '24' hours and '0' minutes.
- Select bot:** A dropdown menu showing 'Extract document information' and a 'Browse' button. An orange arrow points to the 'Browse' button.
- Preview bot:** A checkbox with a magnifying glass icon.
- Run bot and dependencies using:** A radio button selected for 'Latest version'.
- Bot priority:** A dropdown menu showing 'Medium'.
- Data privacy tag (optional):** A text box with a '(x)' icon.
- Input values:** A section with a checked checkbox for 'sDocumentID'. Below it, a text box contains the expression '\$ProcessFile.output[sDocumentID]\$', with an orange arrow pointing to it and another orange arrow pointing to the '(x)' icon to its right.

Small text notes are present: 'Used only as internal reference during design and must be unique within this process' for the Element name field, and 'If Bot does not start within the specified time, this Task will timeout. Maximum is 24 hours, minimum is 1 minute.' for the Bot queue timeout section. Another note at the bottom states: 'Used only as an additional hidden field that can be searched for Data Privacy purposes. Variables will be populated at runtime'.

Likewise, we need to re-check the address using the Google Map address validation API.

After the Bot Task we just added, add an API Task from the left-side menu.

First, enter both the Element name and the Task name as "ValidateAddressAgain".

Then, to the right of "Select API Task", press the browse button, from here press browse in the top right and select "Validate Address".

Check both sKey and sPropertyAddress. For sKey, click the X and, under Process, choose Request.

Because this Process started with a form, we choose this option for values from that form. The variable type is Input and choose API Key [TextBox0]-STRING.

This is the value from the text box you added in the form.

For the sPropertyAddress, click the X and, under Task, choose GetDocumentInfoAgain, then choose Output and select the variable GetDocumentInfoAgain.sPropertyAddress.

Task: API

Executes an API call with inputs and outputs

Element name
ValidateAddressAgain

Used only as internal reference during design and must be unique within this process

Task display name
ValidateAddressAgain (x)

☐ Hide this task from users

Select API Task
Validate address Browse

[Preview API Task](#)

Run bot and dependencies using
☒ Latest version

Data privacy tag (optional)
(x)

Used only as an additional hidden field that can be searched for Data Privacy purposes.
Variables will be populated at runtime

Input values

☒ sKey

(x)

☒ sPropertyAddress

(x)

Finally, we will add a Bot Task after the API task we just added.

Although this duplicates the last Bot Task on the “IF” side, because the source of the data is different, we will add another node.

For both the Element name and Task name, type “EnterValidatedDataToWebpage”.

Once again to the right of Select Bot, press Browse, from here select browse in the top right and select “Enter data to webpage”.

Check all the Input values. Each of the variables will be from the “GetDocumentInfoAgain” except for the sPropertyAddress which comes from the “ValidateAddressAgain” node.

Add the following variables to their respective fields:

Variable	Node variable
----------	---------------

sLandlordName	GetDocumentInfoAgain.output[sLandlordName]
sMonthlyRent	GetDocumentInfoAgain.output[sMonthlyRent]
sPropertyAddress	ValidateAddressAgain.output[sPropertyAddress]
sTenantName	GetDocumentInfoAgain.output[sTenantName]
sTypeOfLease	GetDocumentInfoAgain.output[sTypeOfLease]

Task: Bot

Executes a bot with inputs and outputs

Element name
 ←

Used only as internal reference during design and must be unique within this process


Task name
 ← (x)

☐ Hide this task from users

Bot queue timeout
 hours minutes

If Bot does not start within the specified time, this Task will timeout. Maximum is 24 hours, minimum is 1 minute.

Select bot
 →

 Preview bot

Run bot and dependencies using
☒ Latest version

Bot priority

Data privacy tag (optional)
 (x)

Used only as an additional hidden field that can be searched for Data Privacy purposes. Variables will be populated at runtime

Input values

☒ sLandlordName ←

(x) ↓

☒ sMonthlyRent

(x)

☒ sPropertyAddress

(x)

☒ sTenantName

(x)

☒ sTypeOfLease

(x)

Finally, select the last node, currently labeled “Failed”.

Choose the new End process status as “Completed”.

Save your Process. Your lab is complete.

Testing

On the lab page, there is a button that says, “Download Leases”.

Save the ZIP file to your computer and extract the five PDF lease documents contained.

Run your Process using the Run button in the upper-left corner.

A new tab will appear with your form.

Select both "Lease Agreement 1" and "Lease Agreement 3" PDF files on your computer.

Switch to your lab tab and copy your API key.

Switch back to the VM tab and paste that into the API key field.
(Hint: Use right-click and Paste.)

Then click the Submit button in the lower-right of the screen.

Processing documents takes about 1 minute. If your document does not need validation, a new web page appears, and the automation enters the data from the extracted document.

Now, one of our documents contains an unexpected value for the monthly payment, it says "TBD", or "to be determined". Since this is not a numeric value, the document extraction sends the document to validation for a human set of eyes to take a look.

Field and Document Rules

Since this is only a single field, we can resolve this with a Field Rule.

Let's return to our learning instance by clicking Manage > Learning Instances.
(The far left-side menu may be rolled up; you can click the > button at the bottom-left of the screen to expand it.)

selectlisdmenu.png

Select the learning instance with your name, then click the Monthly Rent field.

Next, click on the Field Rules tab above Field name and click Add Rule.

Click on Field Rule 1 to expand it.

This presents us with an IF... THEN construct for the value of the field. For the condition, leave the comparison as "equal to", and in the text box below that, enter TBD.

In the THEN section, leave the field as Monthly Rent, and change the Action Type to "set value" and in the text box below that, enter the number 0 (zero).

fieldrules.png

What this does is change the value of TBD for the Monthly Rent to 0 (zero). These rules can be set for specific fields, or, by using Document Rules, can be used to modify another field or even perform calculations.

After you make this change, click Update.

Switch back to Automation from the far-left menu, open your Lease document process by clicking on its name, and run the process again with the problematic document.

Extra Credit

If you still have some extra time, let's talk about how we can speed up the data entry into the form. In its current, erm, form, we use the Recorder: Capture action to fill in each field. Let's consider other methods.

There are two ways we can do this: One is to send keystrokes to the web page, and one is to use JavaScript injection to set the values of the web objects directly.

If you choose to send keystrokes, you will need to make sure your starting place is correct. Using a single Recorder: Capture, position the cursor in a text box by using the "Click" subaction... or you can use the "Set text" subaction to put text in every field. How? By using tab characters (not "\t", use \$\$String:Tab\$\$) to jump from field to field.

What about the radio buttons and Submit button? Those can be triggered with a space character, just like when navigating the page with a keyboard only.

You can also use the Simulate keystrokes action to do the same thing, but you must watch out that you're starting in the correct field.

With JavaScript injection, you first must validate the site does not use CSP (Content Security Policy) programming or the injection will fail. Once validated, you must create the JavaScript dynamically in a text file because JavaScript will not allow for use of Automation Anywhere variables.

Incorrect:

```
document.getElementById("textbox1").value = $$sUserName$$;
```

Correct:

```
document.getElementById("textbox1").value = "George Jones";
```

Once written to a text file (the Log to file action is perfect for this), use the Browser: Run JavaScript action to execute the script. This allows for full-speed manipulation of the web page with any JavaScript actions you wish.

Copyright 2022 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.